RECEIVED
CENTRAL FAX CENTER

FEB 2 0 2007

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Applicant:	John L. Reid	Art Unit:	2195
Serial No.:	09/756,579	Examiner:	Lewis Alexander Bullock, Jr.
Filed:	January 8, 2001	Docket:	ITL.0463US P9817
For:	Sharing Classes Between Programs	Assignee:	Intel Corporation

Mail Stop **AF**
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

SUPPLEMENTAL DECLARATION UNDER RULE 131

Sir:

I, Timothy N. Trop, state as follows:

1. I am the attorney who prosecuted the present application and prepared the application.

2. On or about November 2, 2006, I submitted a Declaration in this case. In paragraphs 4 and 5 I referred to an invention disclosure. A true and correct copy of that disclosure is attached hereto.

3. The disclosure shows a date of July 2000 and shows receipt by the Intel legal team on July 12, 2000.

Date of Deposit: February 20, 2007
I hereby certify that this document is being facsimile transmitted to the United States Patent and Trademark Office (Fax No. 571/273-8300) on the date indicated above.
Cynthia L. Hayden
Cynthia L. Hayden

4. Under the heading "Description" attached to the disclosure, there is a complete description of each of the elements of the claims. Specifically, the duplication of member data is described at the top of page 4, first few lines. Enabling to the application to define an address space of shared memory specific to each application is described at the same place. Sharing a class is described throughout the disclosure.

5. I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and, further, that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Date: 2/20/2007 Timothy N. Trop

P9817- 15088

INTEL INVENTION DISCLOSURE

ATTORNEY-CLIENT PRIVILEGED COMMUNICATION

DATE: 10 Jul. '00

JUL 10 2000

It is important to provide accurate and detailed information on this form. The information will be used to evaluate your invention for possible filing as a patent application. When completed and signed, please return this form to the **Legal Department at JF3-147.** If you have any questions, please call **264-0444.**

1. Inventor: Reid (Last Name) John (First Name) L. (Middle Initial)

Phone (503) 264-8870 M/S: JF1-32 Fax #

Citizenship: U.S. WWID: 10060003 Contractor: YES ____ NO ✓

Inventor E-Mail Address: john.l.reid@intel.com

Home Address: 16893 NW Waterford Way

City Portland State OR Zip 97229 Country U.S.

*Corporate Level Group (e.g. IABG, NCG, CEG) ____ Division CPD Subdivision ____

Supervisor* Ed Jacks WWID 10021321 Phone (503) 712-1196 M/S: JF1-32

Inventor: ____ (Last Name) ____ (First Name) ____ (Middle Initial)

Phone ____ M/S: ____ Fax # ____

Citizenship: ____ WWID: ____ Contractor: YES ____ NO ____

Inventor E-Mail Address: ____

Home Address: ____

City ____ State ____ Zip ____ Country ____

*Corporate Level Group (e.g. IABG, NCG, CEG) ____ Division ____ Subdivision ____

Supervisor* ____ WWID ____ Phone ____ M/S: ____

***If you are unsure of this information, please discuss with your manager.**

(PROVIDE SAME INFORMATION AS ABOVE FOR EACH ADDITIONAL INVENTOR)

2. Title of Invention: Object-based Interprocess Communication Mechanism

3. What technology/product/process (code name) does it relate to (be specific if you can): Communication between two applications on a single computer using a object-based IPC mechanism.

4. Include several key words to describe the technology area of the invention in addition to # 3 above: OOP

5. Stage of development (i.e. % complete, simulations done, test chips if any, etc.): 75%

6. (a) Has a description of your invention been, or will it shortly be, published outside Intel:

NO: ✓ YES: ____ If YES, was the manuscript submitted for pre-publication approval? ____

IDENTIFY THE PUBLICATION AND THE DATE PUBLISHED: ____

(b) Has your invention been used/sold or planned to be used/sold by Intel or others?

NO: ✓ YES: ____ DATE WAS OR WILL BE SOLD: ____

RECEIVED

JUL 12 2000

PATENT DATABASE GROUP
INTEL LEGAL TEAM

ATTORNEY-CLIENT PRIVILEGED COMMUNICATION

(c) Does this invention relate to technology that is or will be covered by a SIG (special interest group)/standard/ or specification?

NO: ✓ YES: _____ Name of SIG/Standard/Specification: _____

(d) If the invention is embodied in a semiconductor device, actual or anticipated date of tapeout? —

(e) If the invention is software, actual or anticipated date of any beta tests outside Intel Not planned.

7. Was the invention conceived or constructed in collaboration with anyone other than an Intel blue badge employee or in performance of a project involving entities other than Intel, e.g. government, other companies, universities or consortia? NO: ✓ YES: _____ Name of individual or entity: _____

8. Is this invention related to any other invention disclosure that you have recently submitted? If so, please give the title and inventors: No.

PLEASE READ AND FOLLOW THE DIRECTIONS ON HOW TO WRITE A DESCRIPTION OF YOUR INVENTION

Please attach a description of the invention to this form, DATED AND SIGNED BY AT LEAST ONE PERSON WHO IS NOT A NAMED INVENTOR, and include the following information:

1. **Describe in detail what the components of the invention are and how the invention works.**
2. ✓ **Describe advantage(s) of your invention over what is done now.**
3. **YOU MUST include at least one figure illustrating the invention. If the invention relates to software, include a flowchart or pseudo-code representation of the algorithm.**
4. ✓ **Value of your invention to Intel (how will it be used?).**
5. ✓ **Explain how your invention is novel. If the technology itself is not new, explain what makes it different.**
6. ✓ **Identify the closest or most pertinent prior art that you are aware of.**
7. ✓ **Who is likely to want to use this invention or infringe the patent if one is obtained and how would infringement be detected?**

***HAVE YOUR SUPERVISOR READ, DATE AND SIGN COMPLETED FORM**

DATE: 7-10-00 SUPERVISOR: Edmund R. Jackson

BY THIS SIGNING, I (SUPERVISOR) ACKNOWLEDGE THAT I HAVE READ AND UNDERSTAND THIS DISCLOSURE, AND RECOMMEND THAT THE HONORARIUM BE PAID

An Object-Based Inter-Process Communication Mechanism

Abstract

It is often necessary or convenient for two software programs running on the same computer to communication and share data. Modern day operating systems provide various low-level mechanisms to support this. Unfortunately such mechanisms are all procedure-based access to shared memory. Since a large proportion of current-day software projects are developed using object-oriented languages and object-oriented paradigms, these operating system provided mechanisms are awkward at best and often complex to implement, test, and maintain because of this mismatch. On Windows operating systems, Microsoft provides one solution called the Component Object Model or COM. While COM does provided an object-oriented way to communicate and share data across the boundries of a programs process address space, it also includes substantial overhead. This comes in the form of both a steep learning curve for developers and in the form of run-time overhead necessary to enable all of the features COM publishes – whether or not the program actually uses them. This has implications on the amount of resources a program uses, it's performance, complexity of debug, maintenance, and future modification.

The invention discussed here is a mechanism which provides developers with a way to easily communication and share data between two programs running on the same computer in a fully object-oriented manner. The specific implementation was done using the C++ language under the Microsoft Windows operating system.

Disclaimer

The mechanism described here is in *raw* form; that is, it is described in terms of the basic building blocks necessary. A commercial implementation of this mechanism would surely hide most of the detail from both developer and user through automatic code generation.

For Whom

This mechanism would be used by developers of computer software. Both those developing IPC shareable class objects and those developing applications which use IPC shareable class objects.

Description

Object-based IPC extends the well known advantages of object-oriented programming to the model in which two programs on the same host communicate with each other. The object-based IPC method described here follows mechanisms familiar in Remote Procedure Calls (RPC) but extends the model to deal with object interfaces as opposed to file-scope procedures.

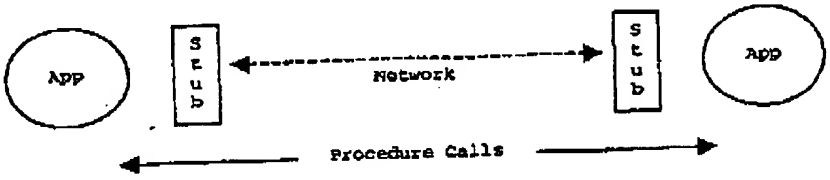

Figure. Existing RPC

The above figure shows the model for existing RPC mechanisms. *Stubs* are generated to hide the fact that a function being called actually exists on another computer on the network. The stubs provide the "glue" necessary to make this as transparent as possible to the application.

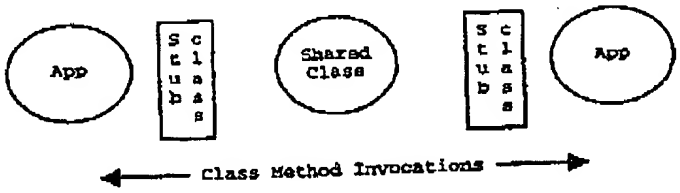

Figure. Object-based IPC

In a vane similar to that in RPC, the object-based IPC mechanism generates *stub classes* which hide the fact that a class being used actually is shared with other applications. Note that while the object-based IPC mechanism being described here could be extended to allow the sharing of classes over a network, for our discussion this is restricted to sharing classes on a single computer.

Mechanics

An implementation of object-based IPC must solve two fundamental requirements:

1. The backing storage for classes is shared memory.
2. All object defined member data which is process specific is duplicated in the address space of all using processes.

The easiest way to describe the implementation is to ignore the stub classes for now. After all, they only exists to "hide the glue".

To share a class we require that it's backing storage come from shared memory. As mentioned previously, operating systems provide rudamentary support for sharing memory between multiple programs on a computer. Under Win32 the preferred method of memory-mapped files can be used.

Normally a class object is instantiated in C++ using the complier defined new operator as follows

```
Class* object = new Class;
```

or implicitly as

```
Class object;
```

In the first case the object's storage is allocated out of the program's default memory heap. In the latter case it is allocated out of either the program's global memory pool (if at file-scope) or out of it's stack memory (if at function-scope).

Neither of these methods satisfy our basic requirement that the object's storage come out of shared memory. To solve this problem we make use of the C++ *placement* new operator. Put simply, this operator allows us to specify the memory which is to be used as backing storage for the class object.

```
Class* object = new(memory) Class;
```

A second issue is that, since the class object is shared, we only want to instantiate one object and share it. To enable this and assure that the backing storage come from shared memory, all of this can be nicely encapsulated in a class object of it's own. The current prototype calls this class Share. Share sets up shared memory using operation system provided services and instantiates the class to be shared out of this memory. Since Share should be independent of the type of class to share, this information is provided to

Share at compile time as a parameterized type using C++ templates. The following example shows the use of Share to instantiate an arbitrary class to share.

```
Share<Class> share;
Class* object = share.instance();
```

Once we have our shared class we must satisfy the second of our requirements – that of assuring that member data within the class which is process specific be duplicated in the address space of each process using the class.

Since our object-based IPC mechanism does not and should not be aware of specific member data in a class object to be shared, we require that shareable class objects provide a method which is invoked to perform this duplication. The current prototype defines an interface called Shareable which class objects implement for this purpose. This interface defines a method call Init (and also an Uninit) which is called prior to using the object. Inside Init, the object duplicates any process specific data and returns a handle. This handle is then provided back to the object in method invocations. The object uses this handle to resolve to the appropriate process specific member data. An example shows this.

```
void* handle = object.Init();
object->method(handle, ...);
```

This is a good example of "glue" that our stub class would hide from an application – namely the calling of Init and the existence of an extra handle parameter.

Summary

Object-based IPC provides a mechanism which allows applications the ability to intercommunication and share data in an object-oriented way.

The advantage of existing IPC mechanisms is that is is object-based, lightweight, and simple to work with/debug. COM provides similar functionality and more. The problem is that the application is burdened with COM's overhead of providing this "more" whether or not it uses it or not. This results in overweight and complex to work with/debug programs. Requirements may indicate that COM is a better choice, but there are clearly applications for which this is not the case – object-based IPC attempts to fill this need.